Comput. Math. Applic. Vol. 15, No. 4, pp. 319–323, 1988
Printed in Great Britain.

0097-4943/88 $3.00 + 0.00

A UNIQUE HIERARCHY ON THE SUCCESSIVE CONDENSATIONS OF A DIGRAPH

MICHAEL J. MANTHEY and KEITH PHILLIPS
Department of Computer Science, Department of Mathematical Sciences and the Computing Research Laboratory, New Mexico State University, Las Cruces, NM 88003, U.S.A.

Abstract—We call a sequence of condensations of a digraph D a hierarchy on D. We define a particular hierarchy based on iterated condensations of directed cycles. When the nodes of D are interpreted as active computational elements and the arcs as communication links between these elements, the hierarchy is a structural model of a concurrent computational system. The cycles are also used to express conservation laws. In this interpretation, locally observable causal relationships, that is, those defined by the function composition of the computational elements, exist only within the levels of the cycle hierarchy.

1. OVERVIEW

Our purpose is to define a type of hierarchy on digraphs. The origin of these hierarchies lies in concurrent, as opposed to sequential, computational systems. A natural consequence of the definition is the uniqueness of this hierarchy on a given digraph. We discuss origins after presenting the definition. We follow in general the notation and terminology of Ref. [1] for graphs and digraphs.

Each step in the process collapses a directed cycle in a digraph to a node in a new digraph, in a prescribed manner. When iterated, the process defines a finite sequence of increasingly simpler digraphs which are structural invariants of the original digraph. These invariants have computational and set theoretic interpretations. Our presentation is set theoretic, and rather formal. This style has the disadvantage of obscurity, the advantage of being easily programmable.

In Section 2 we give a precise definition of the cycle hierarchy in set theoretic and graph theory terms. This is the main contribution of the paper. In Section 3 we discuss models of concurrent computational systems and how cycle hierarchy can be used to describe them analytically. This material is dependent on the discussion by one of us in Ref. [2]. In Section 4 we make some brief conjectures on the universality of the cycle hierarchy.

2. THE DEFINITION OF CYCLE HIERARCHY

Let D_0 be a digraph with node set V and arc set A. Let X_i be a sequence of partitions of V. A *hierarchy* h on D_0 is an ordered sequence of condensations $D_{i+1} = X_i(D_i)$ induced by X_i, whose initial term is X_0 (D_0). We will define a particular hierarchy h_{cyc} which is induced by the cyclic subgraphs of D_0. We begin now the definition of cyclic subgraphs of D_0. The first part of the construction is purely combinatorial.

Let $V = \{1, 2, \ldots, n\}$, and let $\{C_1, C_2, \ldots, C_k\} = \Gamma$ be a family of nonvoid subsets of V. We will interpret V as the node set of D_0, and Γ as the set of all its nontrivial cycles. Let U be the index set $\{1, 2, \ldots, k\}$. To begin the definition of the cyclic subgraphs, we define three families of subsets of the index set U: Ω_1, Ω_2, and Ω.

Definition of Ω_1, Ω_2 and Ω

A subset J of U is in Ω_1 if and only if $C_i \cap C_j = \emptyset$ for all pairs i, j from J, $i \neq j$.

A set J in Ω_1 is in Ω_2 if it is maximal; that is, if $J \subseteq K \subset U$, then there are elements i and j in K for which $C_i \cap C_j \neq \emptyset$. Clearly the family of sets Ω_2 is uniquely defined. Let Ω be those members of Ω_2 which have the maximum number of elements among the sets in Ω_2. That is, Ω is the set of members J of Ω_2 for which the cardinality $|J|$ is maximal among the elements of Ω_2.

The next step in the definition of h_{cyc} is to choose chains of overlapping sets C_i and C_j from different families of sets indexed by Ω. It is convenient to list the sets of Ω, say

$$\Omega = \{J_1, J_2, \ldots J_p\}.$$

Let

$$I = \bigcup_{k=1}^{p} J_k, \quad \Delta = \{C_j : j \in I\}$$

be the corresponding family of sets from the original family Γ. That is, Δ is the subfamily of Γ consisting of the sets C_i for which i is in I.

We now take Γ to be the set of cycles of D_0. Hence, the C_i are cycles. We first partition Δ into sets of connected cyles. Define a relation R on $\Delta \times \Delta$ as those (C, D) in $\Delta \times \Delta$ for which there is a chain of elements (A_j), $0 \leqslant j \leqslant q$ of Δ satisfying

$$A_0 = C, \quad A_q = D \quad \text{with} \quad A_j \cap A_{j+1} \neq \emptyset \quad (0 \leqslant j < q).$$

It is clear that R is an equivalence relation on Δ, and so uniquely partitions Δ into families of pairwise disjoint sets.

The partition R can also be described in the following terms. Define a graph G having as its nodes the set I. Then (i, j) in $I \times I$ is an edge in G if $i \neq j$ and $C_i \cap C_j \neq 0$. The equivalence relation R on Δ corresponds to the connectedness equivalence relation in G, so the components of R correspond to the connected components of G. There are, of course, well known algorithms for finding the connected components of a graph.

Since each equivalence class is a union of cycles, it follows that the subdigraph consisting of the component is strongly connected. A *cyclic subgraph* of D_0 is one of these component subgraphs.

We now begin the definition of the hierarchy h_{cyc}. For each $i \in I$, let V_i be the node set of C_i. Let

$$S = V - \left[\bigcup_{i \in I} V_i\right].$$

Then $\{V_i : i \in I\} \cup S = \Pi$ is a partition of V. (The elements of S are singletons in this partition.) Let D_1 be the digraph having node set $I \cup S$ (relabeling may be needed). Thus all the nodes in V_i are *condensed* to the node i. Then (i, j) is an arc of D_1 if an arc joins an element of $V_i \ldots$ or i, if $i \in S$) to an element of V_j (or j, if $j \in S$). The digraph D_1 is the *condensation* of D_0 induced by Π. Note that the digraph D_1 has at least one less cycle than D_0.

We have defined a unique process, say P, which can be applied to any digraph D_0 to obtain a digraph $P(D_0)$ which has fewer nodes and fewer cycles than D_0, if D_0 has cycles. If D_0 has no cycles, then it is clear that $P(D_0) = D_0$.

The *cycle hierarchy* of D_0 is the finite ordered set

$$h_{cyc} = (D_i) \quad 0 \leqslant i \leqslant m,$$

where $D_{i+1} = P(D_i)$, D_i has cycles if $i < m$, and D_m has no cycles. The hierarchy is completely and uniquely determined by the original digraph D_0. The process P is well defined from digraphs to digraphs and reduces the number of cycles monotonically.

Some examples of the hierarchy induced by the R-subgraphs of Δ are shown in Fig. 1.

3. COMPUTATIONAL INTERPRETATION

In this section we discuss computational implications of the hierarchy defined in Section 2. The content and notation are dependent on previous work of one of the authors; see Ref. [2].

Let each of the nodes of D_0 be an *actor* (computer + fixed program), and the arcs of D_0 be a fixed set of directed communication arcs between these actors over which flow finite bit strings called *messages*. An *event* is the receipt of a message by an actor. The arrowhead at the end of an arc in D_0 denotes all events occurring at the so-designated actor. A *process* is a connected sequence of events.

Time is defined as the 1–1 mapping of the events constituting a given process onto the integers. As a consequence, every process represents its own sequential time frame. Arcs, then, have no *time* associated with them—*time* is a process, not actor-net, property.

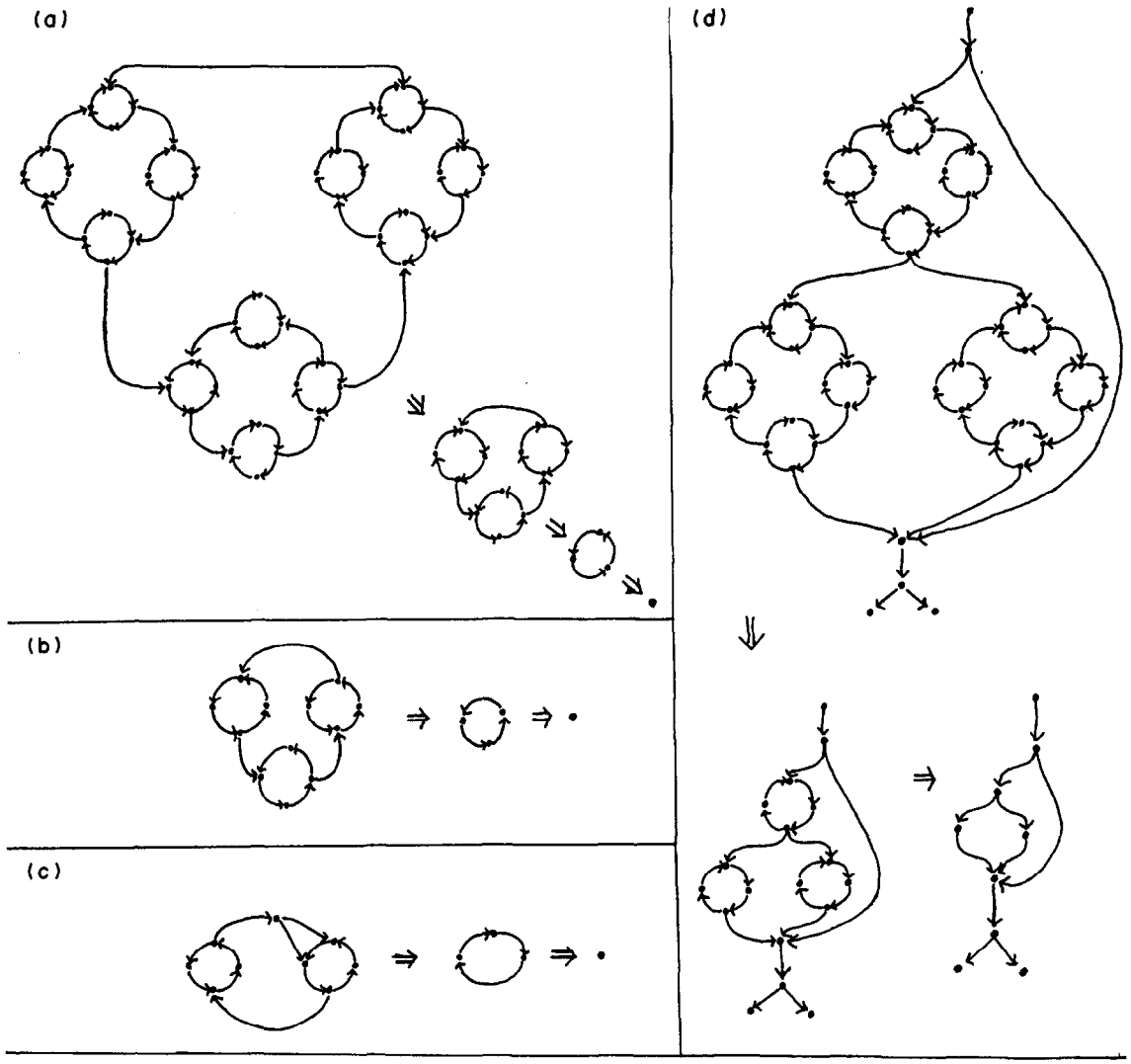
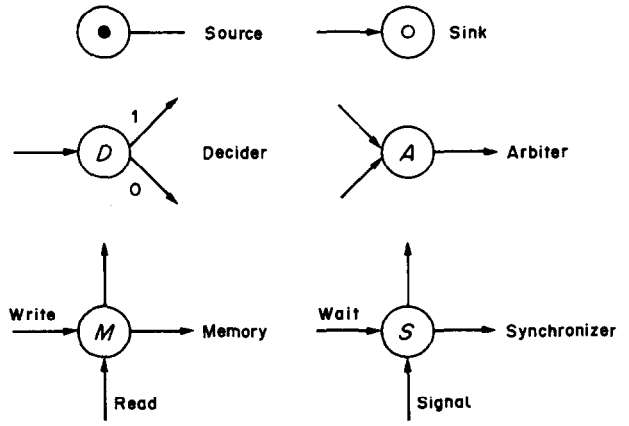


Fig. 1

Another consequence of these definitions is that any given message contains the instantaneous state of some associated process. Notice that the digraph representation of a system does not reflect the number of messages/processes in it. The latter is rather unspecified, and would be supplied from without if, for example, a simulation were to be performed using the network as a basis.

Consider a digraph built from the following (computationally universal) primitive actors:



Sources and sinks are the entry and exit points for messages, and define the boundary of the system represented by the graph with its environment. Sources emit messages acausally and unpredictably from the point of view of the system.

Deciders correspond to "if-then-else", and direct messages onto one of the two out-legs based on message content. Arbiters "arbitrate" so-arriving messages on their two in-legs to the out-leg in a *fair* way, that is, the probability is 0.5 that a message arriving on a particular in-leg is the first one out relative to a co-arriving message on the other in-leg.

Memories are like computer memories in their use, but unlike these, hold but a single bit and are destructive readout (DRO); that is, when read, the bit present is taken by the "reading process", leaving the memory empty. Reading an empty memory is a null operation.

Synchronizers represent the acquisition by a "waiting process" of a permission "stick" from the synchronizer. Signallers deliver "sticks"; that is, release the access permission which they currently hold for use by another waiter. Synchronizers thus force an event in one process to be after an event in a second process, thereby tying the two time frames together at the point of synchronization.

In so doing, synchronizers provide the means of achieving *mutual exclusion* on the use of some *resource* which is guarded by the synchronizer. The mutually exclusive use of (e.g.) tape drives in a computer system by various jobs is typically accomplished using synchronizer-based mechanisms.

Notice that if the "stick" representing a resoruce disappears from the system, that resource is (generally speaking) no longer available to any other process. When a resource (permission "stick") is always returned for re-use, we can state that the resource [and the "stick(s)" representing it] is conserved. Such resource conservation is expressed by *resource invariants* of the form

$$c = c_0 + \Sigma\ \textit{acquisitions} - \Sigma\ \textit{releases},$$

where c_0 is the initial number of sticks, and c the number currently available. Observe that *resource conservation can only occur on cyclic structures in* D_0.

Such resource sharing and process-process synchronization only occurs in concurrent systems; i.e. systems with more than one process. Such systems are inherently indeterminate; i.e. the sequence of events and the final state of a given process is *in principle* not predictable from initial conditions. Hence resource invariants (and flow invariants, which arise from memories in a way similar to synchronizers) represent the only islands of conceptual stability when one is trying to understand the workings of a concurrent computer system.

The cycle hierarchy h_{cyc} represents a *conceptual hierarchy* on the structure of a concurrent system, and once constructed is a stable decomposition of the system into simpler systems. The uniqueness of the hierarchy generated by h_{cyc} means that this structure is an objective property of the system, and thus that all observers of the system must arrive at this same structure in a correct analysis.

In this computational interpretation of h_{cyc}, we can therefore say that

(1) from the point of view of describing computations abstractly, the h_{cyc} hierarchy is *non-procedural*; i.e. not based on function composition, the hierarchy relation for sequential systems. Rather, h_{cyc} captures part-whole relationships;
(2) resource conservation is an *emergent property* of concurrent systems; that is, the whole is greater than the sum of its parts, since conservation is a group phenomenon not isolatable to any single particular process action. Non-determinism is the fundamental emergent property of concurrent system, and arises out of communication over shared unsynchronized memory;
(3) h_{cyc} faithfully captures and preserves the causal relationships between nodes/processes at each level, just with decreasing detail.

4. A PHYSICAL INTERPRETATION

There is nothing in the basic definition of event and process, nor in the computational primitives suggested, that restricts them to the description of computational systems. For example, think of the protons and neutrons in a helium atom as processes, and the mesons they exchange as messages. The resource invariants in this context express the conservation of the quantum numbers that make

a helium nucleus a helium nucleus. The Pauli and other exclusion principles reflect similar resource/mutual exclusion-based relationships (see references).

Similar considerations show that h_{cyc} captures the atom → molecule → macro-molecule → organelle → cell → organ → organism → species/society of our world in a way that function composition cannot. But this should not be a surprise: after all, our world **is** concurrent!

5. COMMENTS ON RELATED LITERATURE

In Ref. [3], a kind of inverse result to ours is reported. Namely, any strong subgraph of a graph can be replaced (non-uniquely) by a circular chain of smaller subgraphs. The paper [2] is a detailed exposition of the motivation for the development of the cycle hierarchy, including an argument that function composition is unsuitable as the hierarchy relation for concurrent systems. The papers [4] and [5] discuss aspects of hierarchies relating to determinism, theoretical physics and computational theory.

Acknowledgement—The authors would like to thank Frank Harary for his help in preparing this paper.

REFERENCES

1. F. Harary, *Graph Theory*. Addison–Wesley, Reading, Mass. (1969).
2. M. J. Manthey, Hierarchy in sequential and concurrent systems. In *Characteristics of Parallel Algorithms*. MIT Press, Cambridge, Mass. (1987).
3. D. R. Knuth, Wheels within wheels. *J. Combin. Theory* **16(B)**, 42–46 (1974).
4. M. J. Manthey and B. M. E. Moret, The computational metaphor and quantum physics. *Communications ACM*, February (1983).
5. M. J. Manthey, Non-determinism can be causal. *Intl. J. of Theoretical Physics* **23**, 10 (1984).